*Survey on trusted execution environment (TEE)*

# Towards building a trusted application environment

Xiao-Feng Li
xiaofeng.li@gmail.com
January, 2020

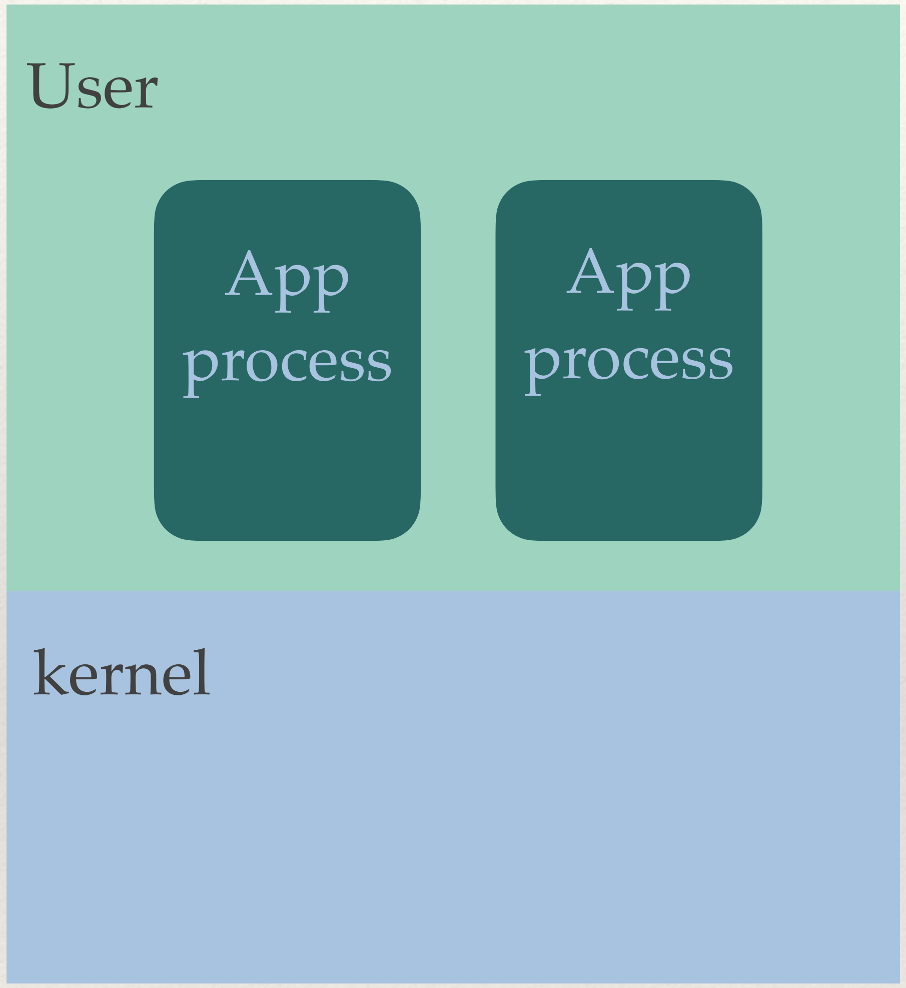
## Agenda

- General security execution environment
  - Secure execution overview
  - Current TEE solutions
  - Comparison and discussion
- TrustZone specific TEE
  - TEE security research
  - TEE usage and innovations

## Computer security

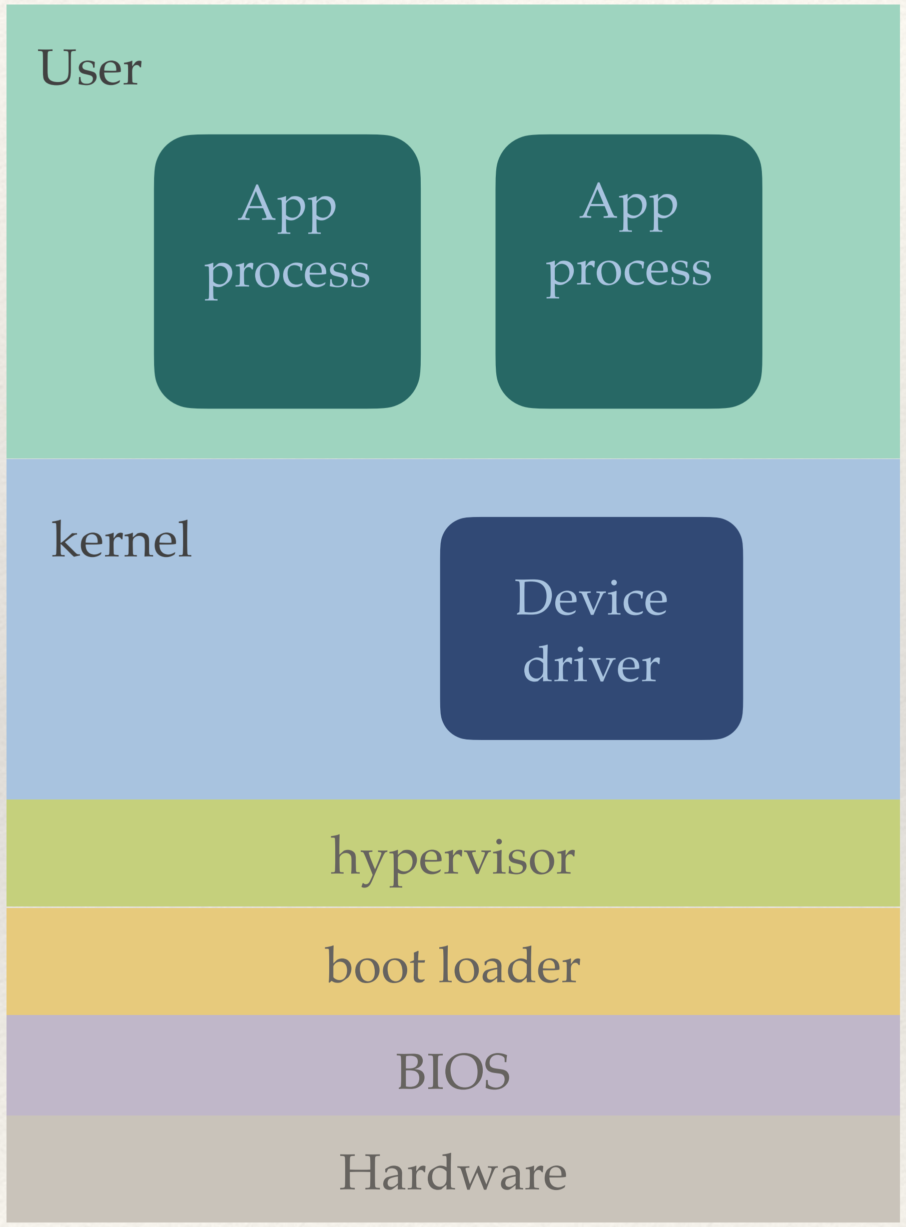
- Info confidentiality
  - Info is not accessible to unauthorized party
- Info integrity
  - Info cannot be modified in an unauthorized or undetected manner
- Attestation and authentication
  - Computing entity can provide verifiable evidence on its state or identity
- Availability

# Typical modern OS



- Process spaces are isolated, with MMU support
- User space has lower privilege, and only accesses resource via sys call, supported by CPU modes
- Kernel makes access arbitration, and does not change frequently
- Resources have user-based access control
- Theoretically well protected

# The reality of OS



- App can be malicious or exploited to access sys call unexpectedly
- Device drivers have kernel privilege and have DMA access
- Every level of the system can be compromised, usually the lower level has higher privilege
- Access to RAM/IO can be snooped
- CPU has side-channel threats

# From defense to prevention

- Community realized software patching not a solution
  - Sandboxing is constructed on top of weak layers
  - Input/output to sandbox are weakly protected
  - Internet makes the defense even harder
- Need to build trustworthy computing
  - Setup the protection from bottom up
  - Executing trusted code, sandboxing, sealing data and communication, remote attestation and provision
  - Business needs for DRM drove the efforts

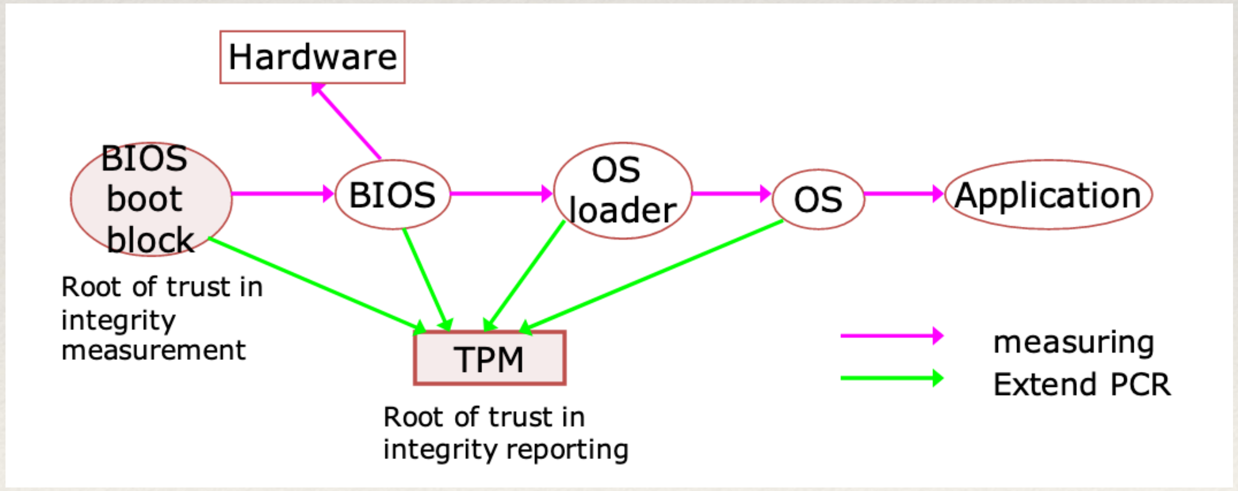
## Root of trust

- The unattested data/code used to establish trust in other components
    - Including initial state+code; optional self-verification
- Common design in hardware
    - A key to sign the state+code, and the provisioned signature
    - The code self-verifies and then measures next component
- Software design is available too
    - Use computation with second pre-image freedom and m-t optimality, based on computer characteristics

https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_10-4_Gligor_paper.pdf

# Starting from TCG for PC

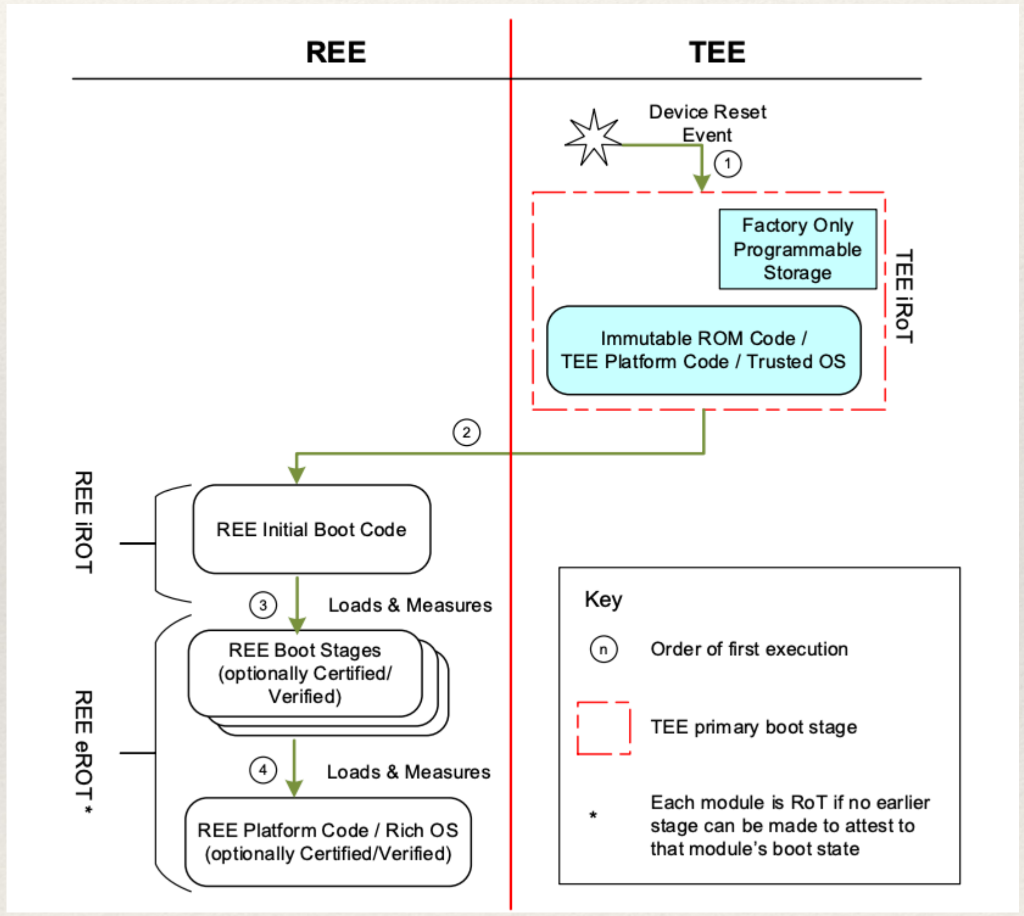
- TPM: a standardized chip with crypto engine, PCRs, NVRAM
  - Root of trust for attestation/storage (EK:EC, SRK)
  - Root of trust for measurement (BIOS boot block)
  - Chain of trust (extend measurements in PCRs)
- Properties:
  - Secure boot, secure storage
  - Attestation and authentication
  - Difficult to apply to the whole and open system


Hardware
BIOS boot block
BIOS
OS loader
OS
Application
Root of trust in integrity measurement
TPM
Root of trust in integrity reporting
measuring
Extend PCR

https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-1-Architecture-01.07-2014-03-13.pdf
TPM defines standardized API so that it can implemented flexibly in HW or SW. Basic APIs: seal/unseal/quote.

# Then GP for mobile device

- GPD TEE: secure environment in main processor.
  - Hardware root of trust (eFuse)
  - Initial boot block in ROM
  - Chain of trust (signed certificates)
- GP vs. TCG
  - Roughly the same goal
  - TEE can run TPM services
  - One focus on mobile, the other on PC/server: converging


REE
TEE
Device Reset Event
1
Factory Only Programmable Storage
Immutable ROM Code / TEE Platform Code / Trusted OS
TEE iRoT
2
REE iROT
REE Initial Boot Code
3
Loads & Measures
REE Boot Stages (optionally Certified/ Verified)
4
Loads & Measures
REE eROT*
REE Platform Code / Rich OS (optionally Certified/Verified)
Key
n
Order of first execution
TEE primary boot stage
*
Each module is RoT if no earlier stage can be made to attest to that module's boot state

https://globalplatform.org/wp-content/uploads/2017/01/GPD_TEE_SystemArch_v1.2_PublicRelease.pdf

## Agenda

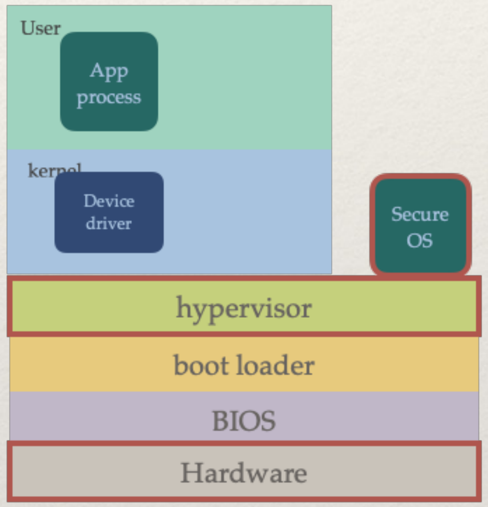
- General security execution environment
  - Secure execution overview
  - Current TEE solutions
  - Comparison and discussion
- TrustZone specific TEE
  - TEE security research
  - TEE usage and innovations

## TEE in general

- An environment that can execute code securely
    - Based on certain root of trust
    - Allow execution of secure applications in protected memory
    - Allow communication between normal apps and secure apps
    - Allow the secure apps to access system resources
    - Support remote attestation/authentication
- There are many implementation variations
    - Normally the TCB  is the smaller, the better

# TEE solution 1: TPM-based

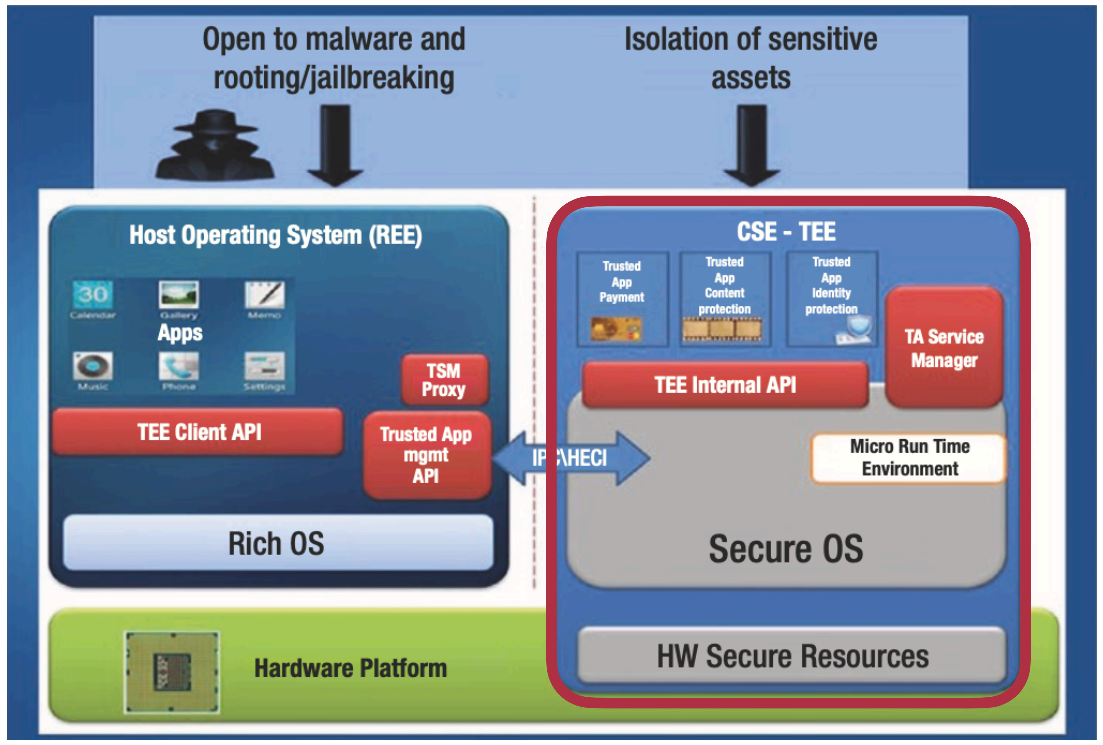
- Extend TPM from static measurement to dynamic measurement
  - OS issues instructions to cleanly start a program, bypassing BIOS/boot loader, etc. Smaller TCB than full static measurement.
  - Chain of trust to hypervisor, OS, and apps
- Examples: Intel TXT, AMD SKINIT
- Any change in the system needs re-provisioning.
  - Suitable for certain scenarios, such as launching hypervisor or small kernel that does not change frequently. Measurement is slow for large program.
  - Only protect program launch, not from runtime attack



# TEE solution 2: coprocessor

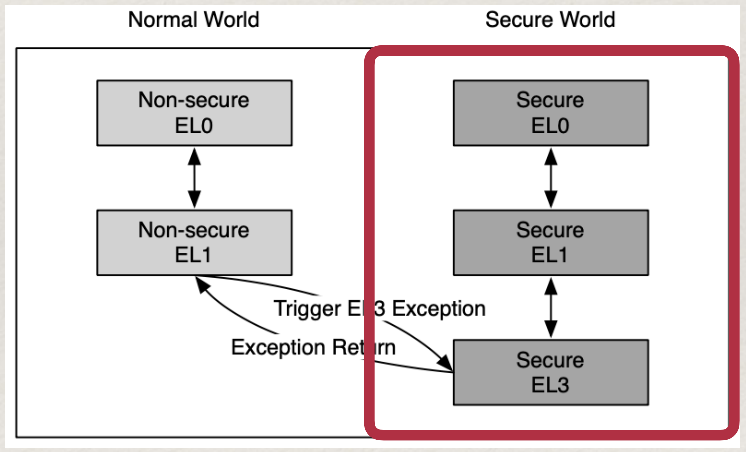
- Usually built-in coprocessor in chipset or SoC
    - With separate OS and apps inside
    - Apps can communicate with external via secure channel
    - Usually includes crypto engine
- Examples:
    - Intel ME, AMD ST, Apple Secure Enclave

- Intel ME can dynamically load Java applet from host with DAL
    - OS is Minix + JVM
    - Access DRAM as secure paging
    - With direct network access


Open to malware and rooting/jailbreaking
Isolation of sensitive assets
Host Operating System (REE)
Apps
TSM Proxy
TEE Client API
Trusted App mgmt API
Rich OS
IPC/HECI
CSE - TEE
Trusted App Payment
Trusted App Content protection
Trusted App Identity protection
TA Service Manager
TEE Internal API
Micro Run Time Environment
Secure OS
HW Secure Resources
Hardware Platform

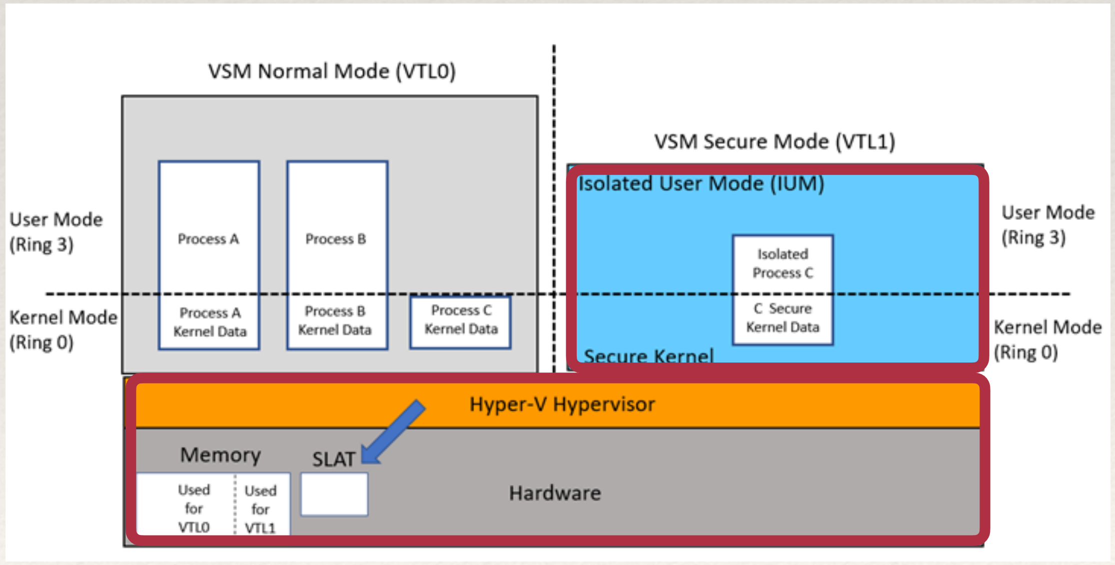
# TEE solution 3: privileged mode

- Use instruction or interrupt to switch between execution environments on same processor
  - Like a special interrupt handler in higher privilege mode
  - Lower mode cannot access its code and memory
  - Can be used to run separate OS and apps
- Examples: Intel SMM, ARM TrustZone
  - Can be used to emulate hardware (inc. TPM)
  - TrustZone extends the support into a full execution environment
    - Supports GP TEE specification



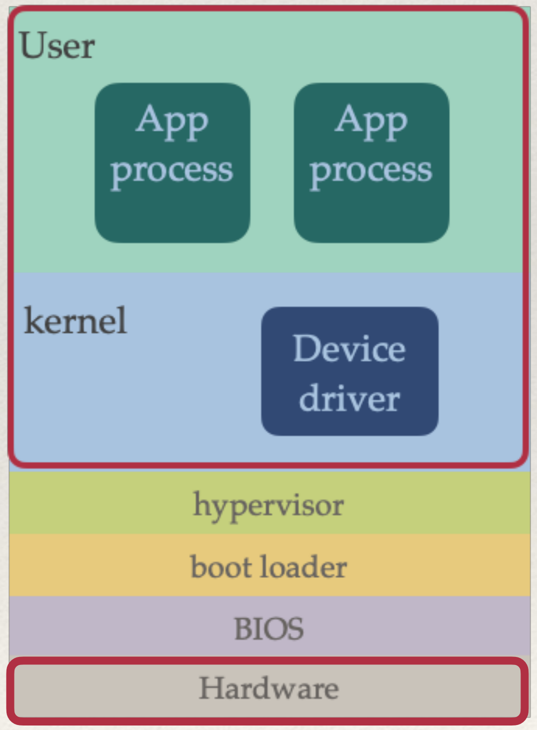
# TEE solution 4: virtualization

- Virtualization can provide a secure VM isolated from the main OS
  - Hardware supports isolation in both memory and IO
  - The secure VM runs security related components, while redirecting sys calls to the main OS where device drivers stay
  - Has to optimize the cost of hypervisor
- Examples: Microsoft VSM, Nova, etc.
  - Secure apps are Trustlets
  - Also implement vTPM
  - Work on X86 and ARM



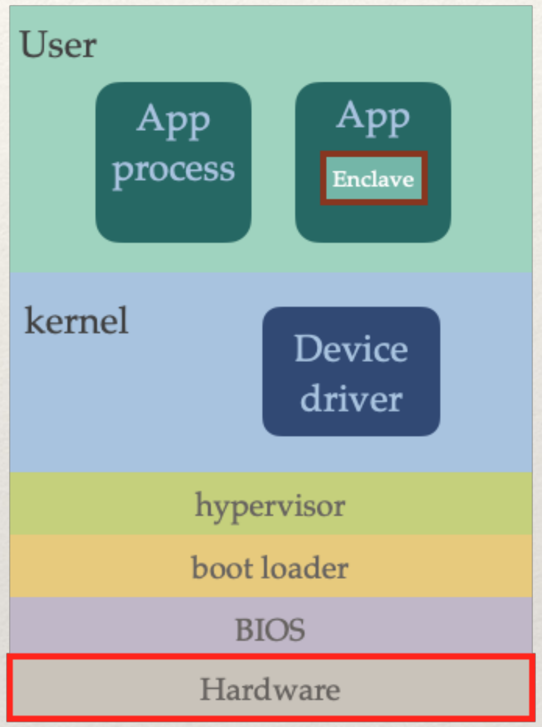
# TEE solution 5: memory encryption

- Encrypt the whole memory content of the OS
  - Protect against physical attacks over bus/DRAM
  - Can be extended to virtualization environment to encrypt a guest OS against compromised hypervisor
    - When the guest obtains its key from tenant
- Examples:
  - Intel TME/MKTME, AMD SME/SEV
  - OS itself is in TCB, not for threats from apps/kernel
  - Mostly useful in cloud, protecting data against CSP



# TEE solution 6: user secure enclave

- Provide memory protection over a specified region
    - Form a secure container within a process
    - The memory is encrypted to/from RAM
    - A minimum TCB inc. only HW+enclave
    - Benefit from all the facilities of the OS
- Examples: Intel SGX, RISC-V Sanctum
    - Initial code is copied from normal space, no secret. Need remote attestation to load/store secrets from outside
    - Does not have local secure I/O.



# Brief comparison

| Solutions | Pros | Cons | Products |
|---|---|---|---|
| TPM-based | Secure boot, secure storage | Assume healthy OS and apps | Intel TXT, AMD SKINIT |
| Coprocessor | Isolated in additional CPU+RAM | Additional core, OS, apps | Intel ME, AMD ST, Apple Secure Enclave |
| Privileged mode | Isolated through CPU mode, one CPU | Cache sharing, static IO partitioning | Intel SMM, ARM Trustzone |
| Virtualization | CPU-ready, flexible I/O partitioning | Hypervisor interposition | Microsoft VSM, Academia: seL4/NOVA |
| Memory encryption | Transparent to OS, against physical attacks | Assume healthy OS and apps | Intel MKTME, AMD SEV |
| User secure enclave | Minimum TCB, against physical attacks | Need remote attestation and provisioning | Intel SGX, RISC-V Sanctum |

# Agenda

- General security execution environment
  - Secure execution overview
  - Current TEE solutions
  - Comparison and discussion
- TrustZone specific TEE
  - TEE security research
  - TEE usage and innovations

# Requirements of a TEE

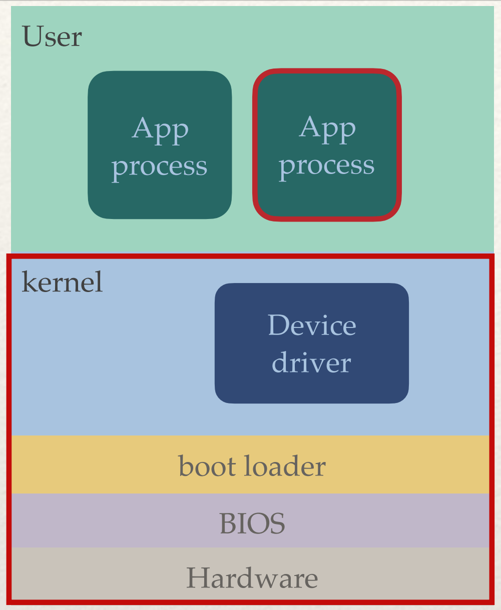
- List of requirements for a TEE system:
  - 1. Trusted app origin and integrity
  - 2. Isolated/protected execution environment
    - Cache, RAM, and I/O. (In other words, partitioned/encrypted)
  - 3. Remote attestation and provisioning
  - 4. Secure persistent data
  - 5. Minimal TCB
  - 6. API and SDK

## Checklist of TEE properties

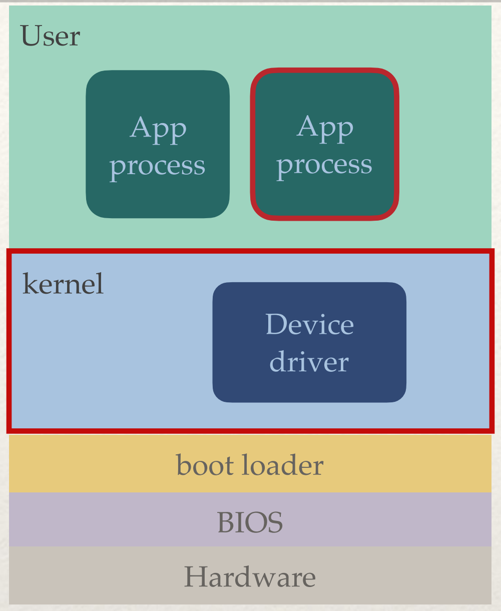
| | Identification and integrity | Isolated and protected Cache | Isolated and protected RAM | Isolated and protected IO | Attestation and provisioning | Secure persistent data | TCB size | API/SDK |
|---|---|---|---|---|---|---|---|---|
| Intel TXT, AMD SVM | Y | Isolated | Isolated | N | Y | Y | Big | Y |
| Intel SGX | Y | N | Y | N | Y | Y | Minimum | Y |
| Apple Secure Enclave, Intel ME | Y | Y | Y | Isolated | Y | Y | Small | Partially |
| ARM Trustzone | Y | N | Isolated | Isolated | Y | Y | Medium | Y |
| AMD SEV, Intel MKTME | Y | Y | Y | Y | Y | Y | Huge | Y |
| Microsoft VSM | Y | N | Isolated | Isolated | Y | Y | Medium | Partially |

*Note, all the solutions assume to work together with certain TPM features for RoT, secure storage

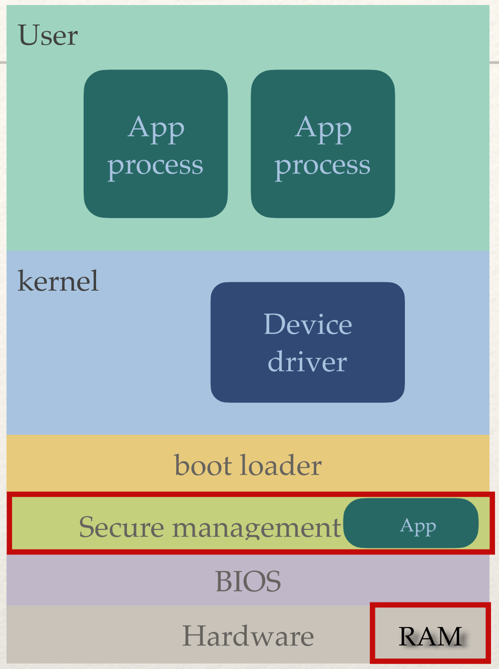
# Isolated/protected environments



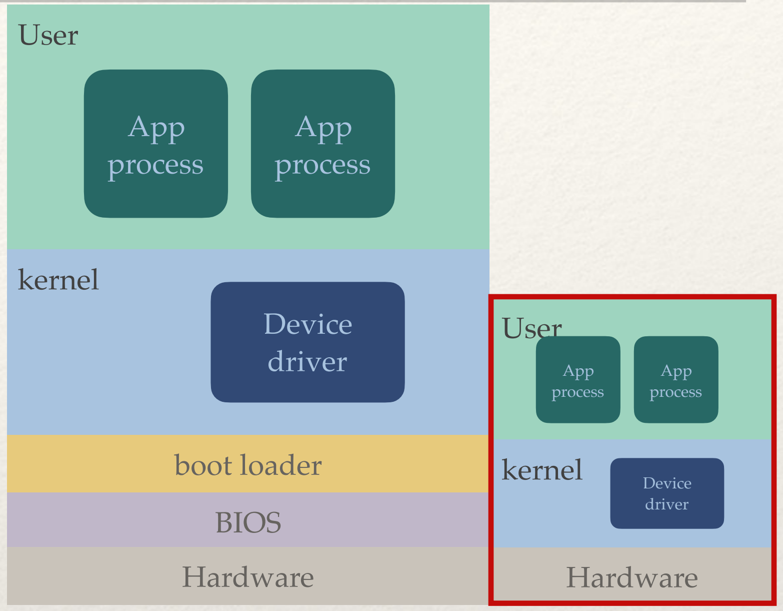
TPM



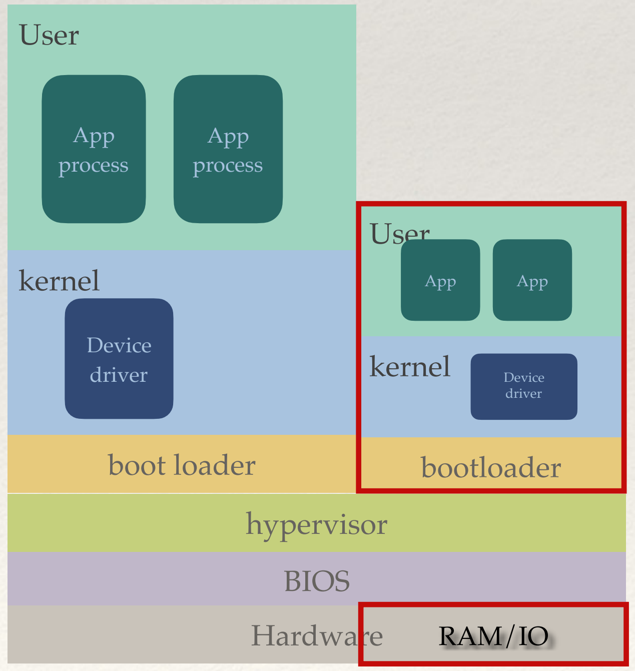
TXT/SVM



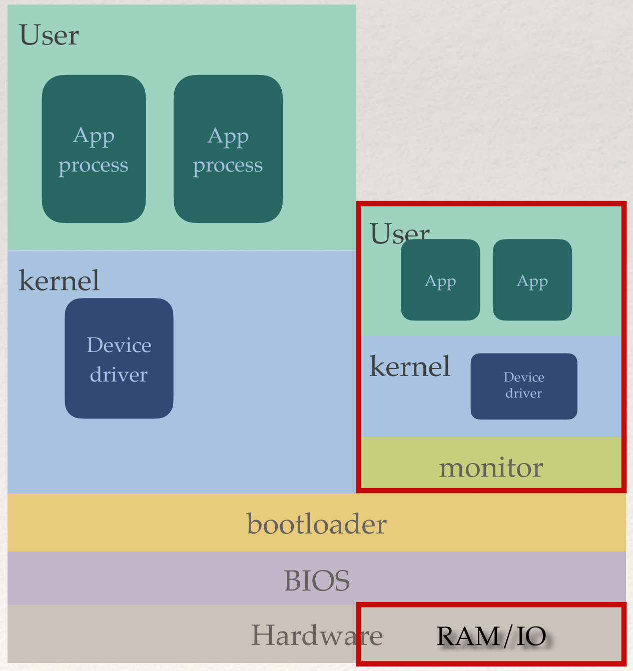
privileged mode



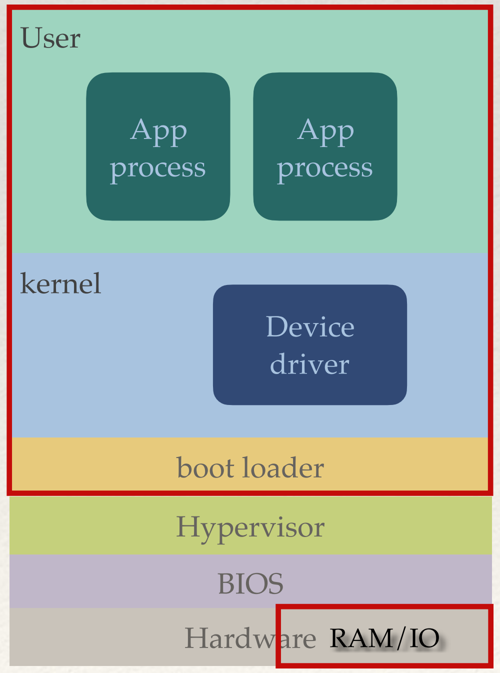
Coprocessor



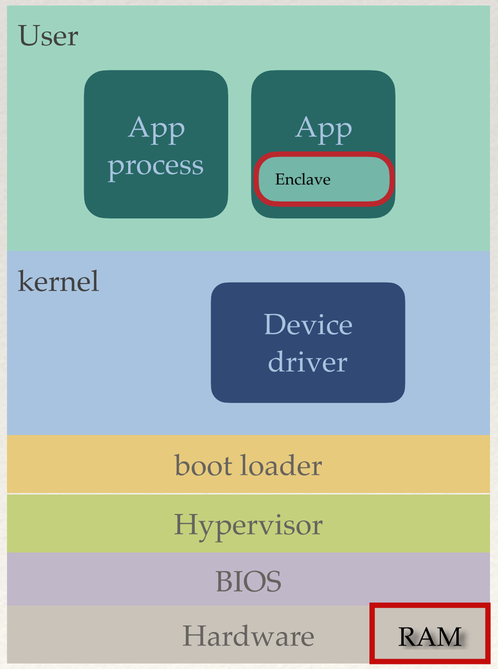
Virtualization



Trustzone

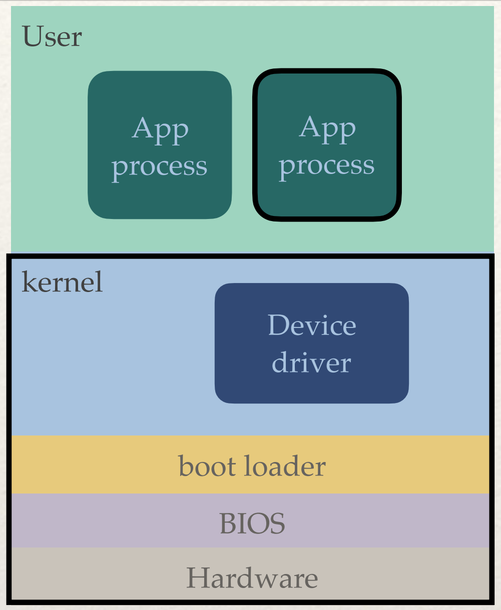


Total mem encryt.

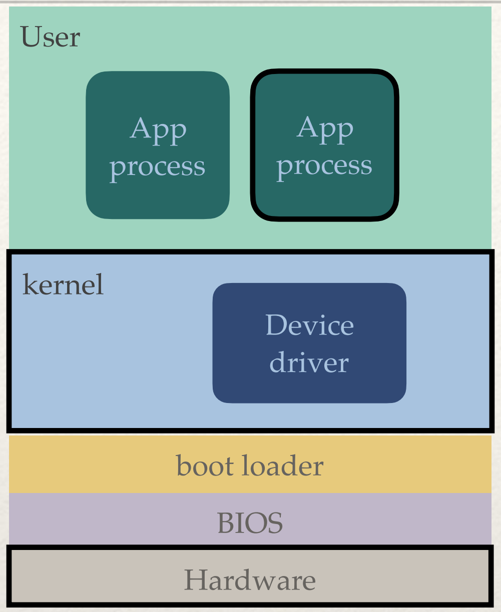


User enclave

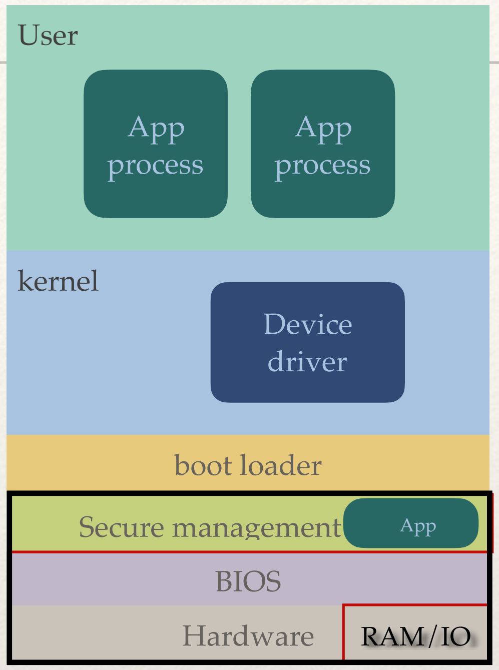
# TCB



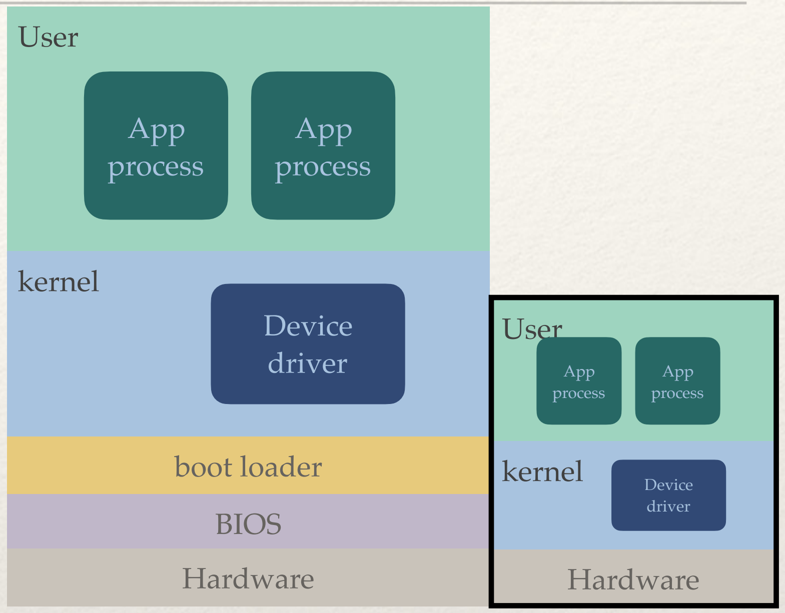
TPM



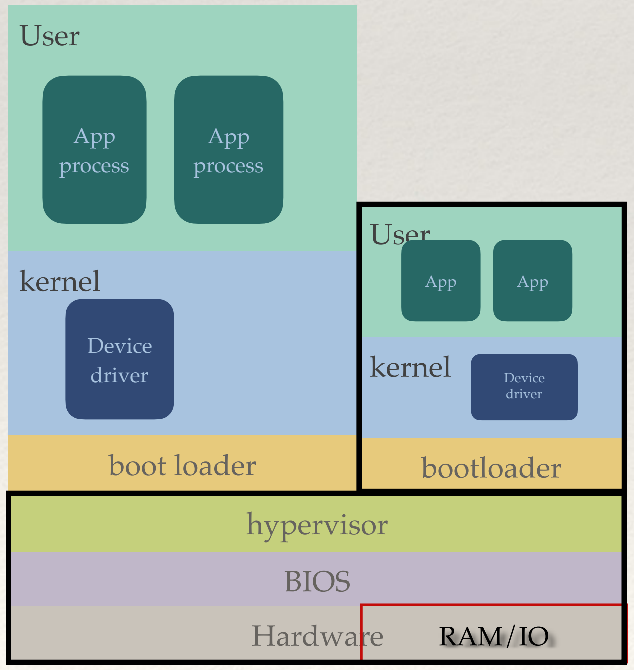
TXT/SVM



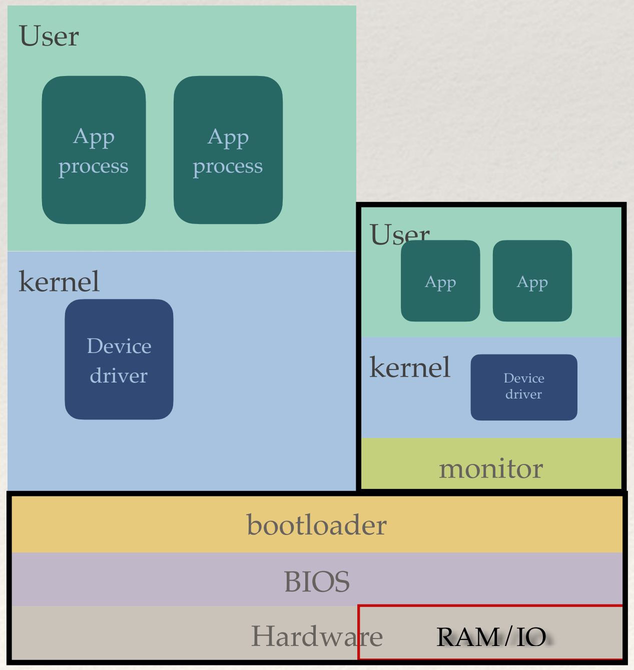
privileged mode



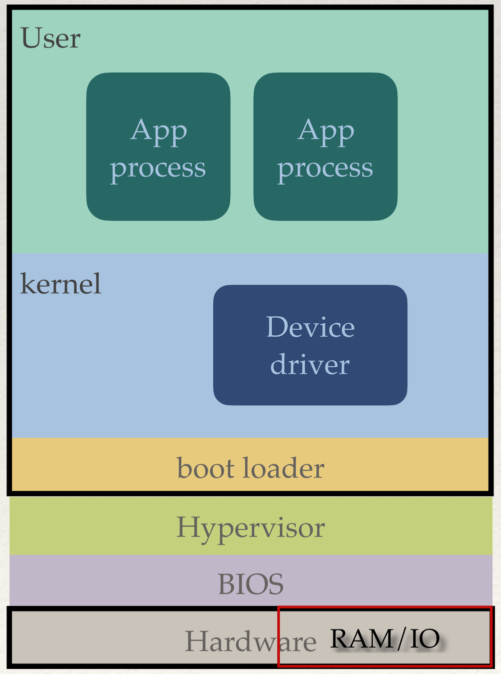
Coprocessor



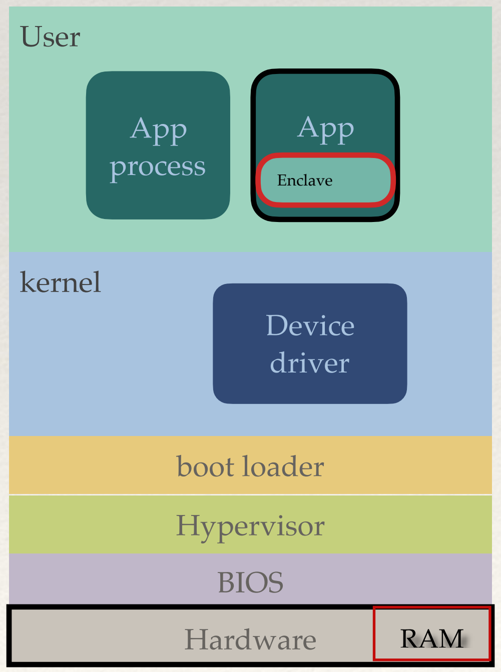
Virtualization



Trustzone



Total mem encryption



User enclave

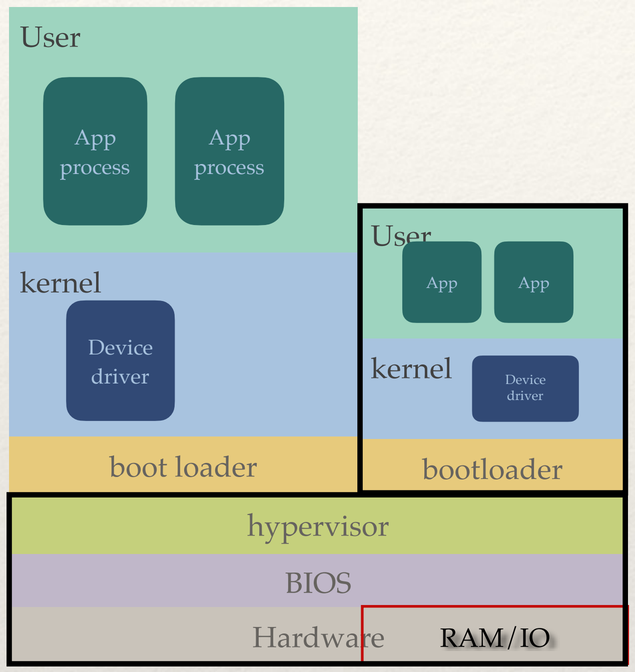
## TEE solution characteristics (1)

- TPM-based solution like Intel TXT, AMD SVM lays a strong foundation towards building TEE by supporting TPM features.
- The requirements for owned devices and shared devices are different. Owned devices may not strongly need total memory encryption like AMD SEV, Intel MKTME, because physical attacks are gated by device locking.
- Secure coprocessor like Intel ME (+TPM) can provide most of the supports, except it does not have good SDK, or flexible IO access. One of the reasons is that coprocessor normally have separate OS/driver support.

## TEE solution characteristics (2)

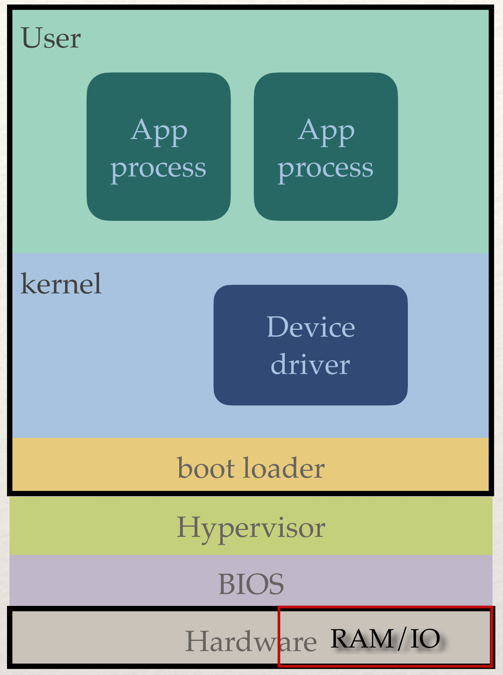
- Virtualization like Microsoft VSM provides flexible RAM/IO isolation to the guest OS. Once combined with total memory encryption, VSM behaves like a full coprocessor by removing the hypervisor from TCB, except it shares CPU with the main OS.
- Privileged mode like TrustZone can be considered as a simplified virtualization, with minimized monitor TCB, and partitioned IO peripherals. It is a cost-efficient solution.
- User-level secure enclave like Intel SGX reduces the TCB to minimum, like a TEE within app. The problem is with IO.

# Potential solution (1)



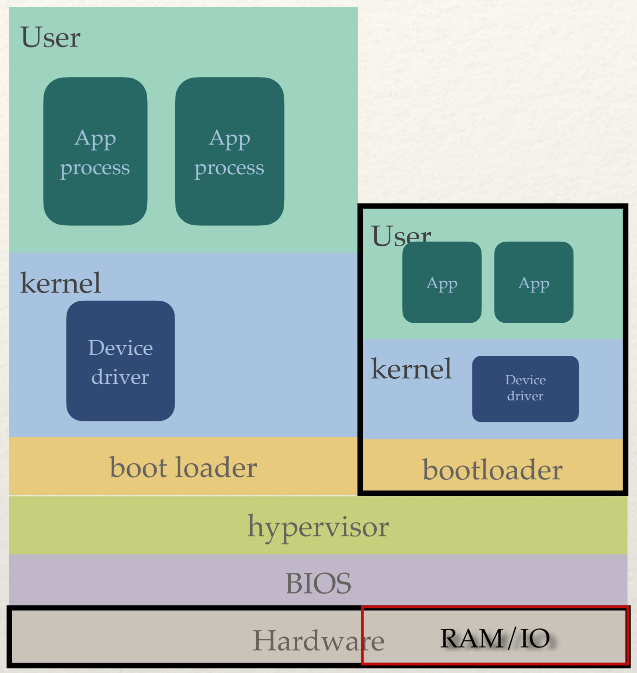
Virtualization

+



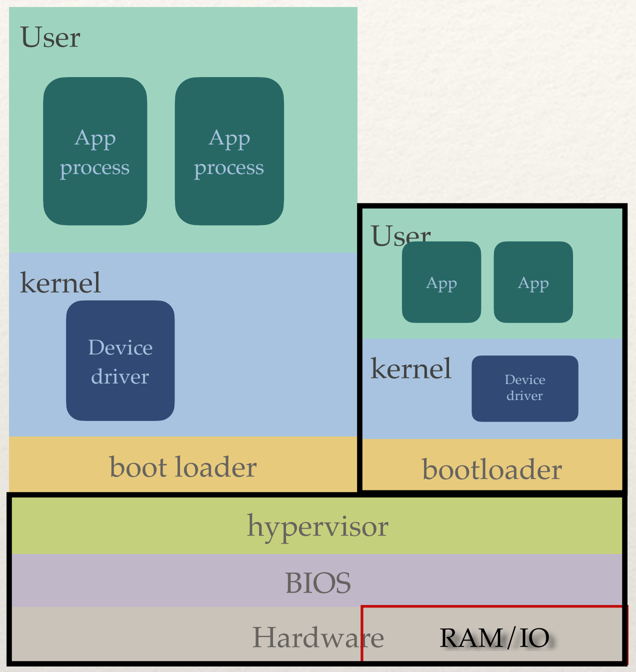
Total mem encryption

=



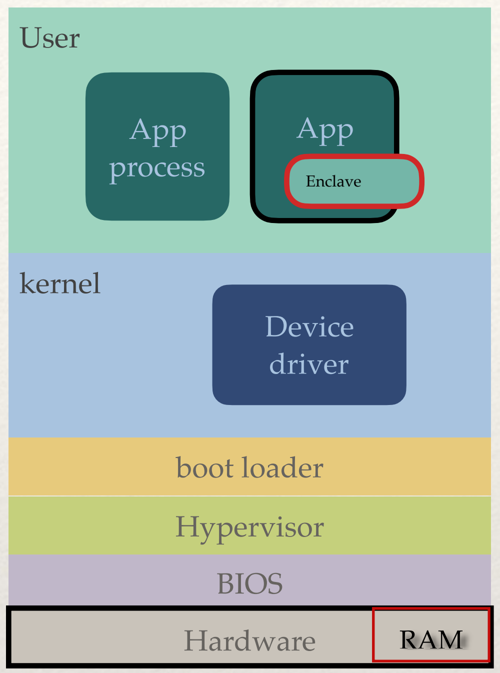
- Benefits
  - Achieve TEE with single processor and medium-sized TCB
  - No special requirement on the hardware beyond the available product

# Potential solution (2)



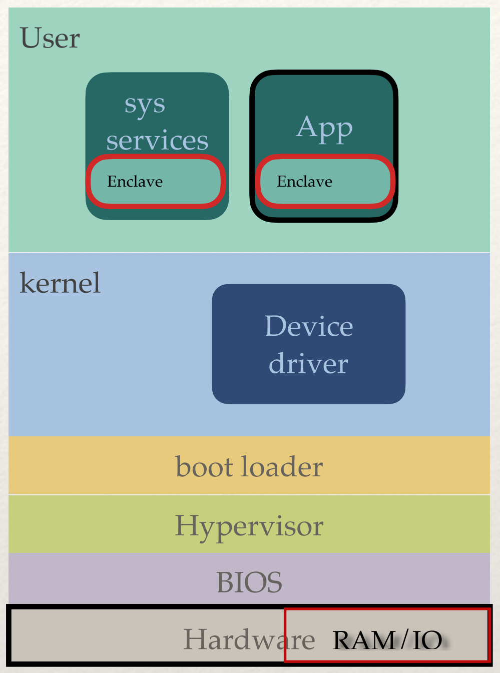
Virtualization

+



User enclave

=



- Benefits
  - Achieve TEE with single processor and minimum TCB
  - Need developing system services that manage the address space of the pre-mapped IO peripherals

## Summary

- TEE is essential to computing system's security.
- The community has built solid foundation with TPM, and then moves forward to propose various TEE solutions that are more practical, reliable and flexible.
- The existing TEE solutions have various pros and cons, and the technologies can be combined to form better solutions, just like how TPM technology is reused everywhere.
- Different computing systems have different requirements on optimal TEE solution. Virtualization and memory encryption are considered most useful for next generation.